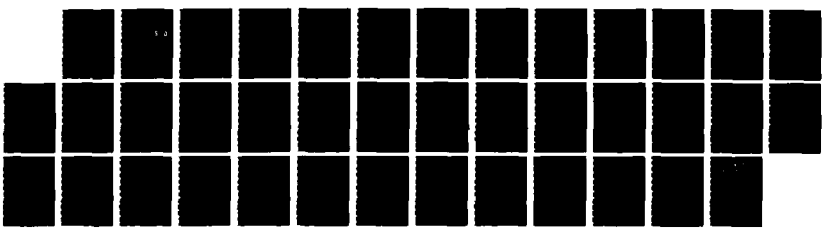
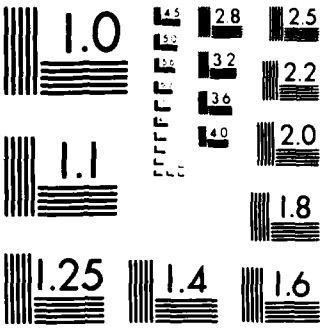
AD-A184 795 THE CIRCULAR SERIAL AUTOCORRELATION COMPUTER PROGRAM 1/1
(U) AMERICAN SOCIETY FOR ENGINEERING EDUCATION
WASHINGTON DC S GILES APR 87 N00014-83-D-0689
UNCLASSIFIED F/G 12/5 NL





(12)

THE CIRCULAR SERIAL AUTOCORRELATION COMPUTER PROGRAM

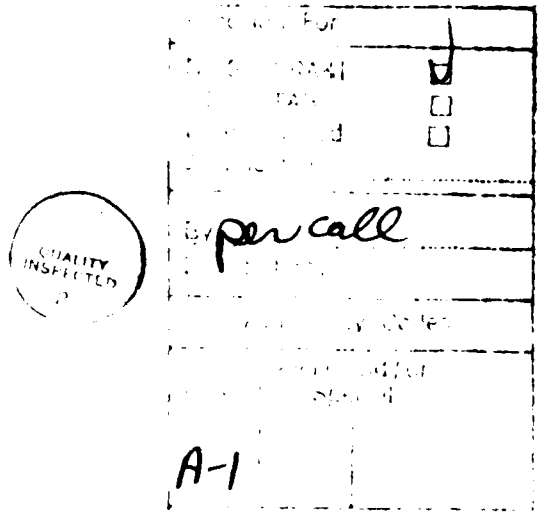
S. Giles

N00014-83-D-0689

April 1987

ACKNOWLEDGEMENTS

This work was supported by the Office of Naval Technology under a poatdoctoral program administered by the American Society of Engineering Education. The author would like to thank the personnel of the Naval Postgraduate School for their kind assistance in this effort, Dr. Gilbert T. Howard and Dr. Sydney R. Parker. The latter is now professor emeritus and is my advisor under the program, without whose help this work would have been impossible. Unfortunately, the author has to thank the personnel at Lawrence Livermore National Laboratory lastly. Special thanks are extended to those at the Labs who have made necessary facilities abundantly available.



INTRODUCTION

This report contains guidelines for using the circular serial autocorrelation filter (CSAF) program. The CSAF program is a later version of an algorithm called SFILTR. In the first section basic things the user of the program should do to run the program are discussed. The requirements of the program are presented in this first section. Interpretation of the program's outputs is given in the second section. In the next section, things which the user should not do are indicated. The fourth section is a three part discussion. Mathematical theory related to the filter is presented in the first part; then the program's organization in terms of functions of its subroutines and main program is discussed in the second part. Following this part are listings of important program variable definitions and the entire program along with a flowchart and sample run streams in the third part. In essence, this report contains the most detailed description of the circular serial autocorrelation filtering tool to date.

1. RUNNING THE CSAF SOURCE PROGRAM

CSAF is a numerical algorithm for filtering serial data utilizing the circularly defined autocorrelation coefficient. Real data to be filtered must be stored in an existing external file named "INPUT" to be list-directed read by the source program. The source program creates an estimate of the input data and creates a corresponding set of error data. In total before execution, there must be three existing files: "INPUT", "OUTPUT", and "FHAT". More discussions of these files are presented in Sections 2 and 3.

CSAF is written in a high level language, FORTRAN. A FORTRAN 77 compatible compiler is needed in forming an executable image. The main program and several subroutines reference several math functions. Thus, the following set of math routines also should be available for double precision applications: sin, cos, arctan, exp, sqrt, and abs. If the above requirements are met, the program can be executed on good personal computers and most mainframes.

It is assumed that a set of parameters related to the input signal's description, the desired error signal statistical test constants, and the desired estimated signal integration method are known. After the execution command the user is prompted to enter six parameters: T, NP, GNT, ZC, NDT, and NFLAG. T is the sampling period of the input signal. NP is the number of samples of the input signal. GNT is the Gaussian noise test constant; it is the critical (alpha) region in statistical theory used in testing the Gaussian nature of the noise signal. ZC is the critical value of the magnitude of the standard normalized Gaussian random variable; it is used in testing the randomness of the error signal. NDT is a logical code for straight line

detrending error signals; its value should be 1 if detrending is desired. NFLAG is a logical code for fast Fourier transforming (FFT) the estimate signal in calculating Fourier coefficients: its value should be 1 if an FFT is desired. (Typical values for GNT and ZC are 0.05 and 1.96, respectively.) After entering the keyboard input parameters, the program should produce three sets of outputs, which are discussed in the following section.

2. INTERPRETING PROGRAM OUTPUTS

Outputs from the source program are written to the terminal screen, into the OUTPUT data file and into the FHAT data file. The terminal displays a set of pertinent statements which provide information on the input, estimate, and error signals. These statements are the following:

1. A statement that the random noise is non-Gaussian at the alpha level, if the test for normality so indicates;
2. An estimate of the signal-to-noise ratio of the input data;
3. The program determined input data's average value (A0) and fundamental angular frequency (W0) along with the inputted sampling period (T), number of data points (NP), remaining keyboard input data; and
4. The even and odd Fourier coefficients (As and Bs respectively) of the estimated signal.

How these printed outputs are calculated is discussed in Section 4. Sample run streams are given on pages 33 through 35.

Real output data are stored in the data files. The OUTPUT data file contains in its first record in sequential order the slope of the straight line detrend (BETA), curve fit error (SE), sampling period (T), fundamental angular frequency (WO), average value (AO) and optimum cutoff harmonic (L-1). Remaining records of OUTPUT contain the even and odd Fourier coefficients. FHAT contains the estimated signal. FHAT's records contain a single value each, and are sequentially ordered from top to bottom, the top of the data file representing the first data point (initial value of the estimate signal). In some applications, either mishandling these output files or the input file presents a problem in running the program. Reads and writes from and to them are important concerns in all applications.

3. AVOIDING MAJOR PROBLEMS

As written, all external files are direct access and must exist before attempting to run the program. Do not fail to create the INPUT, OUTPUT and FHAT files. The program assumes real values. Use of complex data will cause problems. INPUT must contain data which are formatted according to list-directed requirements. Do not place more than one entry per record and neither separate entries by commas nor spaces. If a subroutine is repeatedly called in a redesigned version of the CSAF program, do not fail to reset the associated variables (of the direct access files affected) to the first records of subroutine files called.

In addition to the above data file handling pitfalls, beware of attempting to filter long data records greater than 499 points, especially if it is known that the data have been poorly sampled. The CPU run time increases with optimum harmonic content of the data, and can be as high as 2-1/2 hours for 300 harmonics of 2000 data point records and as low as 10 seconds for 20 harmonics of 128 data points. Aliasing cannot be completely removed by the filter. Aliasing produces cyclical components which the statistical method assumes are not present and the program cannot remove through the subtraction of only a finite number of harmonics.

Other concerns are the following: 1) the number of input data points must be greater than 75 for asymptotic characteristics of the correlation statistics, 2) disk storage for CSAF should accommodate 24 blocks (12288 bits), 3) the Gaussian noise test constant should not be zero because the chi-square tests for normality will not converge, and 4) the critical z-value should not be less than 0.1 for most practical problems. More explanations of how the program works follow, and should give further clarifications of why the above limits are necessary for good program execution.

4. UNDERSTANDING THE PROGRAM

4.1 BASIC THEORY

Given a set of noise contaminated data, the objective of the filtering routine is to recover the continuous desired signal without thereby introducing any jump discontinuities in the recovered signal or its derivatives. Since the desired signal f(t) often contains useful or

understandable information, it is often referred to as the intelligence signal in the sequel. Because of the possible presence of random noise components in discrete recorded data, it is usually not possible to entirely recover the intelligence, and one must be content with an estimate of the sampled version of this signal, $\hat{f}_k$. One can determine an intelligence estimate by many methods. In this work the estimate signal is determined indirectly by construction of a simulation model which can be used to produce a sufficiently continuous signal. The model chosen is a simple truncated Fourier series whose fundamental frequency and coefficients are determined from the input data. Assuming that jump discontinuities of the noise contaminated continuous time domain signal are due to the presence of random noise components, the filtering objective can be met by utilizing the intelligence estimate to get rid of most random noise effects. This can be done if one can define and isolate the noise signal or its estimate.

Assuming that the original input data represents an evenly sampled signal whose fundamental period is the data length (N_p-1) times the sampling period (T), i.e.,

$$\nu_o = (N_p-1)\ T \tag{1}$$

$$\omega_o = 2\pi\ \nu_o \tag{2}$$

one can write a closed expression (model) for the intelligence estimate for L harmonics at every data point k as the following summation:

$$\hat{f}_k\Big|_L = A_0 + \sum_{\ell=1}^{L_0} [A_\ell \cos(k\ell\omega_0 T)] + B_\ell \sin(k\ell\omega_0 T)], \qquad 1 \le L \le L_0 \tag{3}$$

where in the above equation $k \in \{0, \ldots, N-1\}$, L_0 is an optimum harmonic limit, and the coefficients are given for $\ell \in \{1, \ldots, L_0\}$ through the discrete integrals

$$A_0 = \frac{1}{T_0} \int_0^{T_0} f(t)\, dt \tag{4}$$

$$A_\ell = \frac{2}{T_0} \int_0^{T_0} f(t) \cos(\ell\omega_0 t)\, dt, \text{ and} \tag{5}$$

$$B_\ell = \frac{2}{T_0} \int_0^{T_0} f(t) \sin(\ell\omega_0 t)\, dt. \tag{6}$$

The integrals are evaluated using a fifth degree closed Newton-codes formula [1, p. 142], [2, eq. 25.4.14]. Alternatively, if the data are sampled fast enough, the coefficients can be obtained through fast Fourier transformation [3, pp. 75-76]. Then, one can define the error signal, having produced both intelligence estimate and original data at each sample point. The error signal is a set of data values which are the differences between the input data values and corresponding intelligence estimate values, i.e.,

$$e_{kL_0} = f_k - \hat{f}_k\Big|_{L_0} \qquad k = 0, \ldots, N-1. \tag{7}$$

Moreover, one can define a family of error signals parametrized by the maximum harmonic content (L) of the intelligence estimate. Because the error signal for L tends to become more random as L increases, the estimate of the random noise component is defined as

$$\hat{x}_k = e_{kL_o} \qquad k = 0, \ldots, N-1. \tag{8}$$

This noise estimate can be removed, and the resulting signal is $\hat{f}_k\big|_{L_o}$ -- as represented by equation 3 with $L = L_o$. In addition $\hat{x}_k$ can be used in calculation of a signal-to-noise ratio estimate for the contaminated signal. This figure of merit is given by the following expression:

$$\text{SNR} = -20 \log \frac{\text{rms}(\hat{x}_k)}{\text{rms}\left(\hat{f}_k\big|_{L_o}\right)} \tag{9}$$

where rms means "root mean square value" of the enclosed quantity.

L_o in the above paragraph is the harmonic at which the curve-fit error signal e_{kL} (as a function of L) becomes random*. In order to test error signals for randomness, their pertinent statistical characteristics are utilized. First, their individual means (m_L) and standard deviations (s_L) are determined; then each rms (e_{kL}) is found by the formula

$$\text{rms}\ (e_{kL}) = \sqrt{m_L^2 + s_L^2} \qquad 1 \le L \le L_o. \tag{10}$$

* L_o is defined to be less than N/2 to prevent violation of the Nyquist sampling rate.

And from these rms values, curve-fit error norm values defined by

$$E_L = \frac{\text{rms } (e_{kL})}{\text{rms } (f_k|_L)} \tag{11}$$

are determined to gauge the "closeness" of the intelligence estimates to the contaminated signal for each L. Secondly, the circular serial autocorrelation coefficient R of each error signal and the Gaussian nature of the noise estimate are determined. For each L-value for cyclically defined N**, R is given by

$$R|_L = e_{N,L}\, e_{0L} + \sum_{i=0}^{N-1} e_{iL}\, e_{i+1,L} \tag{12}$$

where the statistic R for large N (N > 75) is normally distributed having first and second moments respectively given by the following [4]:

$$\mu|_L = \frac{S_1^2 - S_2}{N-1} \tag{13}$$

$$\sigma^2|_L = \frac{S_2^2 - S_4}{N-1} + \frac{S_1^4 - 4S_1^2S_2 + 4S_1S_3 + S_2^2 - 2S_4}{(N-1)\,(N-2)} - \frac{(S_1^2 - S_2)^2}{(N-1)^2} \tag{14}$$

where $S_i (i=1, \ldots, 4)$ is the ith moment about the origin of the error signal. Thus

---

** N is the number of input data points plus an extra point giving f(NT) = f(0).

$$z\Big|_L = \frac{R\Big|_L - \mu}{\sigma} \tag{15}$$

is normal with zero mean and unit variance. If the calculated z-value magnitude for a given L is greater than 1.96, a null hypothesis of randomness can be confidently rejected at the 5% level. Though the smaller the value of $|z|$ the more confident one can be of randomness, in most practical problems $|Z|$ is not zero; but may be much less than 1.96. Various critical $|z|$-values can be used for various levels of test. 1.96 is simply a popular one corresponding to a 5% critical region. Having $|z|$-value for each L, the task is simply to find a valley turning point on the $|z|$ vs. L curve below the critical z-value, below which value one cannot reject a hypothesis of randomness.

Now, assuming that the fist random error has been determined at $L = L_o$, the next thing to do is to determine whether the hypothesis that the resulting noise is Gaussian is true, statistically. This can be done by performing a chi-square test for normality (in the Gaussian sense). In this work the dynamic range of the noise estimate $(\hat{x}_k)$ is divided into 18 equally spaced intervals (groups); then the theory of grouping for goodness of fit is applied [5, pp. 85-90.] The cumulative normal distribution Q_N for standard normalized noise (z_k)*** is determined using a fifth degree polynomial -- no relationship

---

*** The z_k variable is $z_k = \dfrac{\hat{x}_k - m_{L_o}}{s_{L_o}}$

to Newton-Cotes -- approximation [2, eq. 26.2.17] which is accurate to order 10^{-8}. Theoretical (t_f) and observed (0_f) frequencies are calculated. The chi-square statistic is determined by

$$\chi^2_{\alpha,G-2}\Big|_{\underset{\sim}{x}_k} = \sum_{f=1}^{G} \frac{(t_f - 0_f)^2}{0_f} \equiv \chi^2, \tag{16}$$

where α is the level of significance, G is the number of groups with entries greater than four, and G-2 is the number of degrees of freedom. One can now determine the area under a chi-square distribution from χ^2 to ∞ using the following relation for ν degrees of freedom [6, p. 193]:

$$\phi\ (\chi^2|\nu) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)} \int_{\chi^2}^{\infty} \lambda^{(\nu/2\ -1)}\ e^{-\lambda/2}\ d\lambda. \tag{17}$$

However, for large $\nu > 30$ it is easier to use the inverse formula approximation for χ_α^2 given either the significance level or ϕ [2, eq. 26.4.18]; and to use Q-series expansions for smaller $\nu > 0$ [2, eqs. 26.4.4 and 26.4.5], i.e.,

$$\chi_\alpha^2 = \nu\ [1 - \frac{2}{9\nu} + (W_\alpha - h\ )\sqrt{\frac{2}{9\nu}}\ ]^3 \qquad \nu > 30 \tag{18}$$

$$Q(\chi^2|\nu) = 2\ Q_N(\sqrt{\chi^2}\ ) + 2Z_N\ (\sqrt{\chi^2}) \sum_{\ell=1}^{\frac{\nu-1}{2}} \frac{\sqrt{\chi^2}^{\,2\ell-1}}{1 \cdot 3 \cdot \cdots \cdot (2\ell-1)} \qquad \nu \text{ odd} \tag{19}$$

$$Q(\chi^2|\nu) = \sqrt{2\pi}\, Z_N(\sqrt{\chi^2})\left[1 + \sum_{\ell=1}^{\frac{\nu-2}{2}} \frac{(\chi^2)^{\ell}}{2 \cdot 4 \cdots 2(\ell)}\right] \quad \nu \text{ even} \quad . \tag{20}$$

χ_α should be greater than χ if the null hypothesis is not to be rejected. The constant h_ν depends on the value of W_α [2, eq. 26.4.15] and W_α is the value of the standard normal random variable corresponding to the α critical region. Q_N and Z_N have usual definitions for normal random variables:

$$Z_N(u) = \frac{1}{\sqrt{2\pi}}\, e^{-(u^2/2)} \tag{21}$$

$$Q_N(u) = \int_{-\infty}^{u} Z_N(t)\, dt. \tag{22}$$

If $Q(\chi^2|\nu)$ is greater than α, the hypothesis of normality is not rejected. Otherwise, it is rejected and the non-Gaussian message is printed out in the computer program.

Finally, trend characteristics of the error signals are treated. Straight line trend and critical movements can be reduced. Standard equations for the abscissa intercept (alpha) and slope (beta) of the data are used and are not repeated. This average value of the error is used to adjust the mean of the intelligence estimate. Cyclical movement is minimized by subtracting from the error signal Fourier harmonics (by design). Thus, the error should meet the hypotheses of Wald and Waltowitz [4] for most finite data. Having now presented the essentials of the theory and actual use of the filter program, its organization and examples of its use are presented next.

4.2 PROGRAM ORIENTATION

The circular serial autocorrelation filtering method basically involves determining the fundamental frequency of noisy data, structuring a Fourier series estimate of this data using the curve-fit error characteristics in optimizing the Fourier series estimate and representing the "intelligence" by a set of Fourier parameters. These basic functions are performed in the main program of CSAF. In the first part of the program the number of data points is defined for circular correlation, and the fundamental period and fundamental angular frequency are computed. Secondly, the Fourier estimate is computed for each L from the returned values of Fourier coefficients. Next, the curve-fit error is calculated for each intelligence estimate. Using the correlation statistic the turning point is then found -- if one exists below the critical z-value. Finally, the set of Fourier parameters and other information describing the input data and noise estimate are printed to the screen. In performing these tasks the main program calls several subroutines. These subroutines are discussed in the following paragraphs.

To begin the discussion, there are two options as indicated by the values of the input integers NDT and NFLAG. If NDT is 1, the DETRND subroutine is called. This routine performs least squares, straight line detrending of the error signals before the correlation coefficient of the noise estimate is calculated. If NFLAG is 1, the FFT subroutine is called and the INTEGR subroutine is not called. The FFT subroutine is that of Bloomfield [3] with only minor adjustments; its function is the calculation of the Fourier coefficients. If NFLAG is not 1, the FFT is not called but the INTEGR is called. INTEGR performs the ordinary numerical integration of the waveform to

compute the Fourier coefficients. The actual integration, however, is performed through a subsequent call to the QUAD subroutine, which routine is not discussed. With NDT and NFLAG inputted, the input data are obtained through a call to the DATAIN subroutine. This routine's function is simply to read INPUT, set the last data point equal to the first data point and obtain input data statistics from a call to SERCOR.

The most frequently called subroutine is SERCOR. This subroutine performs two statistical functions, depending on the value of its fifth argument. If this argument is 2, the subroutine simply calculates the mean and standard deviation of the signal appearing as its first argument. If the fifth argument is not 2, the subroutine also computes the circular serial autocorrelation of the signal and the standard normalization of this statistic. There are three other subroutines which calculate statistics of signals. The GAUSS subroutine finds the complement of the cumulative probability [Q_N] of a normal random variable using a polynomial approximation. The QSEXP subroutine finds the value of the area under a chi-square probability distribution from the χ^2 value to infinity [$Q(\chi^2|\nu)$] using the proper series expansion for $Q(\chi^2|\nu)$. TNORM performs the subgrouping of the noise estimate and calculates the corresponding chi-square random variable. Using information from GAUSS and QSEXP, TNORM tests the hypothesis that the noise estimate is random at the GNT level of significance. Finally, the TURN subroutine writes data to OUTPUT and FHAT. More information on how the program functions can be obtained from the flowcharts of the following section (Figs. 1 through 3).

4.3 PROGRAM LISTINGS

Following is a list of important variable definitions of the main program. Also included in the alphabetical list are important variables of the TNORM subroutine, distinguished from the main program variables by asterisks. This list is not complete but should make reading the source program easier.

A	Even Fourier coefficients, an array.
ALP*	Critical region for normality test.
ALPHA	Abscissa intercept of straight line fit of error signal.
AVH	Average value of input data's estimate.
B	Odd Fourier coefficients, an array.
BETA	Slope of straight line fit of error signal.
CHIS*	Chi-square statistics of noise estimate.
CNP	Cumulative normal probability of standard normalized noise estimate, an array.
CX*	Intermediate value of chi-square statistics of noise estimate, an array.
DOF*	Degrees of freedom for noise estimate chi-square statistic.
DX*	Width of groups of making up noise estimate signal.
DDX*	Discrete boundary of DX groups, an array.
E	Error norm, an array.
EAVG	Average value of error signal.
ERR	Error signal, an array.
ESIG	Standard deviation of error signal.

FN	Input data, an array.
FH	Intelligence (Fourier) estimate, an array.
FQ*	Cumulative chi-square probability of noise estimate chi-square statistic.
GNT	Gaussian noise test constant.
LMAX	Maximum number of harmonics calculated.
N	Number of points in circular data.
ND*	Number of sections into which the range of noise signal is divided.
NDT	Logical code for straight line detrending of error signal.
NFLAG	Logical code for performing FFT.
NP	Number of input data points.
NR	Logical code for calculation of correlation coefficient of data.
OF*	Observed frequency of values of noise estimate within DX limits, an array.
Q*	Cumulative normal probability of standard normalized noise.
SAH	Average value of input data's estimate.
T	Sampling period of input data.
TF	Theoretical frequency of values of noise estimate within DX limits, an array.
TRF	Theoretical relative frequency of noise estimate values, an array.
WO	Fundamental angular frequency of input data.
X	Detrended error signal, an array.
X*	Noise estimate signal, an array.

XI Imaginary part of input data (zero) and odd Fourier coefficient in FFT return, an array.

XR Real part of input data and even Fourier coefficient in FFT return, an array.

Z Standard normalized correlation coefficient.

Z* Standard normalized noise estimate values, an array.

ZC Critical magnitude of Z-value.

ZX Z-values of family of error signals, an array.

A copy of the entire source program is given on pages 19 through 26. Flowcharts for the main program and the SERCOR and TNORM subroutines are given on pages 27 through 32. Sample run streams are given on pages 33 through 35.

5. CLOSING REMARKS

We have attempted to answer some fundamental questions concerning the computer program used to low-pass filter data based on the serial correlation coefficient of curve-fit error. Though we have addressed the program's inputs, outputs and principles of operation, there can never be a complete description of the intricacies of efficient utilization of the program. This happens after using the program a number of times and applying variations of the basic theory on which the algorithm is based to practical problems. We have done this using various experimental and simulated data and we have not been able to find a case where the program has not done what it is designed to do. We welcome all comments on the performance of the program from all users.

REFERENCES

1. Burden, R. L., J. D. Faires, and A. C. Reynolds, Numerical Analysis, 2nd ed., Boston, MA: Prindle, Weber & Schmidt Publishing Company, 1981.

2. Abramowitz, M. and J. A. Stegun, Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, Applied Mathematics Series, New York: Dover Publications, 1965.

3. Bloomfield, P., Fourier Analysis of Time Series: An Introduction, New York: Wiley Interscience, 1976.

4. Wald, A. and J. Wolfowitz, "An Exact Test for Randomness in the Non-Parametric Case Based on Serial Correlation," Annals of Mathematical Statistics, Vol. 14, 1943, pp. 378-388.

5. Crow, E. L., F. A. Davis and M. W. Maxfield, Statistics Manual with Examples Taken From Ordinance Development, New York: Dover Publications, Inc., 1960.

6. Freund, J. E., Mathematical Statistics, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1962.

CSAF SOURCE PROGRAM

```
C     THIS PROGRAM USES SERIAL CORRELATOR METHOD TO FILTER INPUT
C     DATA.  FOURIER COEFFICIENTS OF FILTERED DATA ARE FOUND UP
C     TO AN OPTIMUM NUMBER OF HARMONICS.  THE SIGNAL TO NOISE RATIO
C     OF THE ORIGINAL INPUT DATA IS APPROXIMATED AND AN ERROR
C     ESTIMATION OF THIS METHOD IS COMPUTED.

      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(250),B(250),X(500),FN(500),FH(500)
      DIMENSION XR(500),XI(500)
      DIMENSION ERR(500),ZX(250),E(500)

C     THIS SECTION READS THE SET OF INPUT PARAMETERS
C     THE INPUT PARAMETERS REQUIRED ARE:
C          T : SAMPLING PERIOD.
C         NP : NUMBER OF SAMPLED POINTS TO BE USED.
C        GNT : GAUSSIAN NOISE TEST CONSTANT.
C         ZC : CRITICAL VALUE OF Z.
C        NDT : DATA DETRENDING CODE.  NDT=1 IF DETREND IS TO
C              BE USED.
C      NFLAG : FFT CODE. IF FFT IS TO BE USED, NFLAG=1.

      PI=4.0*DATAN(1.D0)
      ZX(1)=0.D0
      WRITE(6,245)
   10 READ(5,*)T,NP,GNT,ZC,NDT,NFLAG
      WRITE(6,260)GNT,ZC,NFLAG,NDT
      N=NP+1
      IF(N.LE.75)WRITE(6,200)
      IF(N.LE.75)GO TO 300
      CALL DATAIN(N,FN,AV,SA)
      IF(NFLAG.EQ.1) GO TO 50
   17 DO 20 I=1,N
   20 ERR(I)=FN(I)
   50 WO=2.D0*PI/((N-1)*T)
      LMAX=(N-1)/2
      LTN=0
      IF(NFLAG.EQ.1) GO TO 30
C      NP=.5D0+2.D0*PI/(T*WO)
C      INT=N/NP
      GO TO 40

C     THIS SECTION DETERMINES THE FOURIER COEFFICIENT OF THE
C     INPUT DATA USING FAST FOURIER TRANSFORM (FFT).

   30 DO 170 I=1,N
      XI(I)=0.D0
  170 XR(I)=FN(I)
      CALL FFT(XR,XI,N-1,0)
      AO=XR(1)
      DO 180 I=1,LMAX
      A(I)=2.D0*XR(I+1)
      B(I)=-2.D0*XI(I+1)
  180 CONTINUE

C     THIS SECTION RECONSTRUCTS THE ORIGINAL INPUT DATA USING
C     FOURIER COEFFICIENT APPROXIMATION AND ESTIMATES THE
C     ERROR VALUES. IF FFT IS NOT USED, A NUMERICAL INTEGRATION
C     METHOD IS USED TO DETERMINE THE FOURIER COEFFICIENTS.

   40 DO 190 L=1,LMAX
      NR=0
      IF(NFLAG.EQ.1) GO TO 60
      CALL INTEGR(T,N,WO,AO,A,B,L,ERR,NP)
   60 DO 100 K=1,N
```

```
      FE=0.D0
      DO 70 I=1,L
      AUG=I*W0*(K-1)*T
   70 FE=FE+A(I)*DCOS(AUG)+B(I)*DSIN(AUG)
      FH(K)=FE+A0
      ERR(K)=FN(K)-FH(K)
  100 X(K)=ERR(K)

C     THIS SECTION USES THE SERIAL CORRELATOR METHOD TO FILTER
C     THE INPUT DATA AND ESTIMATES THE S/N RATIO.  FOURIER
C     COEFFICIENTS OF THE FILTERED DATA ARE OUTPUTTED.

      CALL SERCOR(FH,N,SAH,AVH,2,AZ)
      CALL SERCOR(ERR,N,ESIG,EAVG,2,EAZ)
      IF(NDT.EQ.1)CALL DETRND(X,N-1,ALPHA,BETA)
  120 CALL SERCOR(X,N,SIG,AVG,NR,Z)
      E(L)=DSQRT((ESIG**2+EAVG**2)/(SAH**2+AVH**2))
      ZX(L+1)=Z
      IF(L.LT.3)GO TO 125
      IF((ZX(L).GE.ZX(L-1)).OR.(ZX(L).GE.ZX(L+1)))GO TO 125
      IF(ZX(L).GE.ZC)GO TO 125
      CALL TNORM(GNT,SIG,AVG,N,X,IFLAG)
      WRITE(6,*)'      SNR <=> ',-2.D1*DLOG10(E(L)),' DB'
      IF(IFLAG.EQ.1)WRITE(6,290)GNT*100.
      CALL TURN(N,L,FH,SEE,T,W0,A0,A,B,ZX,BETA,ALPHA,LTN)
      GO TO 195
  125 CONTINUE
      SEE=E(L)
  190 CONTINUE
  195 CONTINUE
      WRITE(6,270)SEE
      IF(LTN.NE.1)WRITE(6,275)
      WRITE(6,207)
      WRITE(6,210)A0,W0,T,N-1
      WRITE(6,215)
      DO 160 I=1,L-1
  160 WRITE(6,220)I,A(I),B(I)
      WRITE(6,250)
  300 CONTINUE
  310 STOP
  200 FORMAT('0','N IS LESS THAN 75.  PROGRAM ABORTS.')
  207 FORMAT('0',/,6X,'A0',11X,'W0',12X,'T',11X,'N',/)
  210 FORMAT(' ',3D13.6,I6,/)
  215 FORMAT('0','HARMONIC',10X,'A',15X,'B',/)
  220 FORMAT(' ',3X,I3,5X,D13.6,5X,D13.6)
  230 FORMAT(' ',D13.6,12X,D13.6)
  245 FORMAT(1X,'ENTER T,NP,GNT,ZC,D-CODE,FFT-CODE',/)
  250 FORMAT(//)
  260 FORMAT('0','GAUSSIAN NOISE TEST CONSTANT =',F6.4,/,
     &1X,'CRITICAL Z-VALUE =',F7.4,/,1X,'FFT-CODE =',I2,/,1X,'D-CODE =',
     &I2,/)
  270 FORMAT('0','RMS CURVE-FIT ERROR =',E12.5,/)
  275 FORMAT('0','TURNING POINT BELOW CRITICAL Z-VALUE WAS NOT FOUND.',
     &//)
  290 FORMAT('0','NOISE IS NON-GAUSSIAN AT THE',F4.1,' % LEVEL.')
      END
C.....
C.....
      SUBROUTINE DETRND(Z,L,ALPHA,BETA)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Z(0:499)
      REAL*8 A,B,C1,C2,DELTA,T1,T2,S1,S2,TS
      REAL*8 X(0:499),Y(0:499)
   10 DO 90 K=1,2
```

```
      S2=0.D0
      S1=0.D0
      T1=0.D0
      T2=0.D0
      TS=0.D0
      DO 20 I=0,L
      IF(K.EQ.1)GO TO 15
      Y(I)=-C2*X(I)+C1*(Y(I)-A)
      X(I)=C1*X(I)+C2*(Y(I)-A)
   15 CONTINUE
      IF(K.EQ.1)X(I)=DFLOAT(I+0)
      IF(K.EQ.1)Y(I)=Z(I)
      T1=T1+X(I)
      T2=T2+X(I)**2
      TS=TS+Y(I)*X(I)
      S1=S1+Y(I)
   20 S2=S2+Y(I)**2
      DELTA=(L+1)*T2-T1**2
      A=(T2*S1-T1*TS)/DELTA
      B=((L+1)*TS-T1*S1)/DELTA
      VM=S1/(L+1)
      SIG=DSQRT((S2-S1**2/(L+1))/L)
      XTY=(L+1)*A-S1+B*T1
      IF(K.NE.1)GO TO 80
      C1=DCOS(DATAN(B))
      C2=DSIN(DATAN(B))
      ALPHA=A
      BETA=B
      GO TO 90
   80 DO 50 I=0,L
      Z(I)=Y(I)
   50 CONTINUE
   90 CONTINUE
  100 RETURN
      END
C....
C....
      SUBROUTINE TNORM(ALP,XSIG,XM,N,X,IFLAG)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION DDX(25),Z(25),X(N),CNP(25),CX(25)
      DIMENSION TF(25),TRF(25)
      INTEGER OF(25)
      XMAX=X(1)
      XMIN=X(1)
      ND=20
      IFLAG=0
      CHIS=0.D0
      CNP(1)=0.D0
      CNP(ND+1)=1.D0
      DDX(1)=-10.D36
      DDX(ND+1)=10.D36
   20 DO 30 I=1,N
      IF(X(I).GE.XMAX)XMAX=X(I)
      IF(X(I).LE.XMIN)XMIN=X(I)
   30 CONTINUE
      DX=(XMAX-XMIN)/(ND-2)
   50 DO 100 I=1,ND-1
      DDX(I+1)=DX*(I-ND/2)+XM
      Z(I)=(DDX(I+1)-XM)/XSIG
      A=Z(I)
      CALL GAUSS(A,Q,AZ,AS2PI)
      CNP(I+1)=1.D0-Q
  100 CONTINUE
  110 DO 120 J=1,ND
```

```
      TRF(J)=CNP(J+1)-CNP(J)
      TF(J)=TRF(J)*N
      OF(J)=0
      DO 118 I=1,N
      IF((X(I).GT.DDX(J)).AND.(X(I).LT.DDX(J+1)))OF(J)=
     &1+OF(J)
  118 CONTINUE
  120 CONTINUE
      IC=0
      IB=0
  130 DO 200 J=1,ND
  132 CONTINUE
      IF((OF(J).GE.5).OR.(J.EQ.ND))GO TO 195
      OF(J+1)=OF(J)+OF(J+1)
      TF(J+1)=TF(J)+TF(J+1)
      IC=IC+1
  140 DO 180 K=1,ND+1-IC-J
      OF(J+K-1)=OF(K+J)
      TF(J+K-1)=TF(K+J)
  180 CONTINUE
      IF((J.GT.ND-IC).AND.((IB+IC).LE.ND))IB=IB+1
      IF(J.GT.ND-IC)GO TO 210
      GO TO 132
  195 IB=IB+1
  200 CONTINUE
  210 DOF=IB-3
  220 DO 240 J=1,IB
      CX(J)=((OF(J)-TF(J))**2)/TF(J)
      CHIS=CHIS+CX(J)
  240 CONTINUE
      CALL QSEXP(DOF,CHIS,FQ)
      AT=1.DO-ALP
      IF(FQ.GE.AT)IFLAG=1
  300 RETURN
      END
C.....
C.....
      SUBROUTINE QSEXP(XNU,XO,FQ)
      IMPLICIT REAL*8(A-H,O-Z)
   10 X=DSQRT(XO)
      CALL GAUSS(X,Q,Z,S2PI)
      NU=XNU+1.D-9
      JEFF=INT(XNU/2.DO)
      MUTT=INT(XNU-2.DO*JEFF+.5DO)
      QSU=0.DO
      ID=1
      IF(MUTT.EQ.0)GO TO 140
  100 DO 120 I=1,(NU-1)/2
      ID=(2*I-1)*ID
  120 QSU=QSU+(X**(2*I-1))/ID
      QXN=2.DO*Q+2.DO*Z*QSU
      GO TO 160
  140 DO 150 I=2,(NU-2)/2
      ID=2*I*ID
  150 QSU=QSU+(X**(2*I))/ID
      QXN=S2PI*Z*(1.DO+QSU)
  160 FQ=1.DO-QXN
  200 RETURN
      END
C.....
C.....
      SUBROUTINE GAUSS(X,Q,Z,S2PI)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION B(5)
```

```
      DATA (B(I),I=1,5) /.319381530D0,-.356563782D0,1.781477937D0,
     &-1.821255978D0,1.330274429D0/
      PI=4.D0*DATAN(1.D0)
      S2PI=DSQRT(2.D0*PI)
      X0=X**2
      Z=(DEXP(-X0/2.D0))/S2PI
      XPP=.2316419D0
      T=1.D0/(1.D0+XPP*X)
      SUM=0.D0
   50 DO 75 I=1,5
   75 SUM=SUM+B(I)*(T**(I))
   80 Q=Z*SUM
  150 RETURN
      END
C.....
C.....
      SUBROUTINE DATAIN(N,F,AVG,SIG)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION F(500)
      OPEN (30,FILE='INPUT.DAT',STATUS='OLD')
      DO 10 I=1,126
   10 READ(30,*) F(I)
      CALL SERCOR(F,N,SIG,AVG,2,0.D0)
      F(N)=F(1)
   80 RETURN
      END
C.....
C.....
      SUBROUTINE INTEGR(T,N,W0,A0,A,B,NFQ,F,INT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION F(N),A(NFQ),B(NFQ),D(500),E(500),U(500),V(500)
      K=NFQ
      U(1)=0.D0
      V(1)=0.D0
      D(1)=F(1)
      E(1)=0.D0
   50 WK=(K)*W0*T
      DO 120 L=2,INT+1
      AUG=WK*(L-1)
      D(L)=F(L)*DCOS(AUG)
   90 E(L)=F(L)*DSIN(AUG)
  120 CONTINUE
      CALL QUAD(IC,INT,T,D,SIA)
      U(INT+1)=SIA
      CALL QUAD(IC,INT,T,E,SIA)
      V(INT+1)=SIA
      A(K)=2.D0*U(INT+1)/(T*INT)
      B(K)=2.D0*V(INT+1)/(T*INT)
      IF(K.GT.1)GO TO 160
      A0=.5D0*(F(1)+F(INT+1))/INT
      CALL QUAD(IC,INT,T,F,A0)
      A0=A0/(INT*T)
  160 CONTINUE
  210 RETURN
      END
C.....
C.....
      SUBROUTINE QUAD(IC,INT,T,Q,SIA)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Q(500)
      SIA=0.D0
      IC=0
      DO 130 I=1,INT-4,4
      IC=IC+1
```

```
      SIA=SIA+2.0D0*T*(7.0D0*Q(I)+32.D0*Q(I+1)+12.D0*Q(I+2)
     &+32.D0*Q(I+3)+7.D0*Q(I+4))/45.D0
  130 CONTINUE
      IH=INT-4*IC-1
      IG=1+4*IC
      IF(IH.EQ.3)SIA=SIA+3.D0*T*(Q(IG)+3.D0*Q(IG+1)+3.D0*Q(IG+2)
     &+Q(IG+3))/8.D0
      IF(IH.EQ.2)SIA=SIA+T*(Q(IG)+4.D0*Q(IG+1)+Q(IG+2))/3.D0
      IF(IH.EQ.1)SIA=SIA+T*(Q(IG)+Q(IG+1))/2.D0
  160 RETURN
      END
C.....
C.....
      SUBROUTINE SERCOR(X,N,SIG,AVG,NR,Z)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(N),S(4)
      R=0.D0
      DO 10 I=1,4
   10 S(I)=0.D0
      DO 40 I=1,4
      DO 40 J=1,N
   40 S(I)=S(I)+X(J)**I
      AVG=S(1)/N
      SIG=DSQRT((S(2)-N*AVG**2)/(N-1))
      IF(NR.EQ.2)GO TO 100
      EXR=(S(1)**2-S(2))/(N-1)
      VAR=(S(2)**2-S(4))/(N-1)-(S(1)**2-S(2))**2/(N-1)**2
      VAR=VAR+(S(1)**4-4*S(2)*S(1)**2+4*S(1)*S(3))/((N-1)*(N-2))
      VAR=VAR+(S(2)**2-2*S(4))/((N-1)*(N-2))
      DO 70 I=1,N-1
   70 R=R+X(I)*X(I+1)
      R=R+X(1)*X(N)
      Z=DABS(R-EXR)/DSQRT(VAR)
      IF(Z.GE.25.D-4)NR=1
  100 RETURN
      END
C.....
C.....
      SUBROUTINE FFT(XR,XI,N,INV)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION XR(N),XI(N),UR(15),UI(15)
      LOGICAL FIRST
      DATA FIRST /.TRUE./
      IF(.NOT.FIRST)GO TO 120
      UR(1)=0.D0
      UI(1)=1.D0
      DO 110 I=2,15
      UR(I)=DSQRT((1.D0+UR(I-1))/2.D0)
  110 UI(I)=UI(I-1)/(2.D0*UR(I))
      FIRST=.FALSE.
  120 IF(N.GT.0  .AND.  N.LE.2**16)GO TO 130
      INV=-1
      RETURN
  130 NO=1
      II=0
  140 NO=NO+NO
      II=II+1
      IF(NO.LT.N)GO TO 140
      I1=NO/2
      I3=1
      IO=II
      DO 260 I4=1,II
      DO 250 K=1,I1
      WR=1.D0
```

```
      WI=0.D0
      KK=K-1
      DO 230 I=1,I0
      IF(KK.EQ.0)GO TO 240
      IF(MOD(KK,2).EQ.0)GO TO 230
      J0=I0-I
      WS=WR*UR(J0)-WI*UI(J0)
      WI=WR*UI(J0)+WI*UR(J0)
      WR=WS
  230 KK=KK/2
  240 IF(INV.EQ.0)WI=-WI
      L=K
      DO 250 J=1,I3
      L1=L+I1
      ZR=XR(L)+XR(L1)
      ZI=XI(L)+XI(L1)
      Z=WR*(XR(L)-XR(L1))-WI*(XI(L)-XI(L1))
      XI(L1)=WR*(XI(L)-XI(L1))+WI*(XR(L)-XR(L1))
      XR(L1)=Z
      XR(L)=ZR
      XI(L)=ZI
  250 L=L1+I1
      I0=I0-1
      I3=I3+I3
  260 I1=I1/2
      UM=1.D0
      IF(INV.EQ.0)UM=1.D0/DFLOAT(N0)
      DO 310 J=1,N0
      K=0
      J1=J-1
      DO 320 I=1,II
      K=2*K+MOD(J1,2)
  320 J1=J1/2
      K=K+1
      IF(K.LT.J)GO TO 310
      ZR=XR(J)
      ZI=XI(J)
      XR(J)=XR(K)*UM
      XI(J)=XI(K)*UM
      XR(K)=ZR*UM
      XI(K)=ZI*UM
  310 CONTINUE
  400 RETURN
      END
C.....
C.....
      SUBROUTINE TURN(N,L,FH,SE,T,W0,A0,A,B,ZX,BETA,ALPHA,LTN)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION FH(N),A(L),B(L),ZX(L+1)
      OPEN (20,FILE='OUTPUT.DAT',STATUS='OLD')
      OPEN (21,FILE='FHAT.DAT',STATUS='OLD')
      LTN=1
      A0=A0+ALPHA
      WRITE(20,*) BETA,SE,T,W0,A0,L-1
      DO 80 I=1,L-1
   80 WRITE(20,*)A(I),B(I)
      WRITE(21,*)(FH(I),I=1,N)
  200 RETURN
      END
```

Figure 1

CSAF

Circular Serial Autocorrelation Main Program

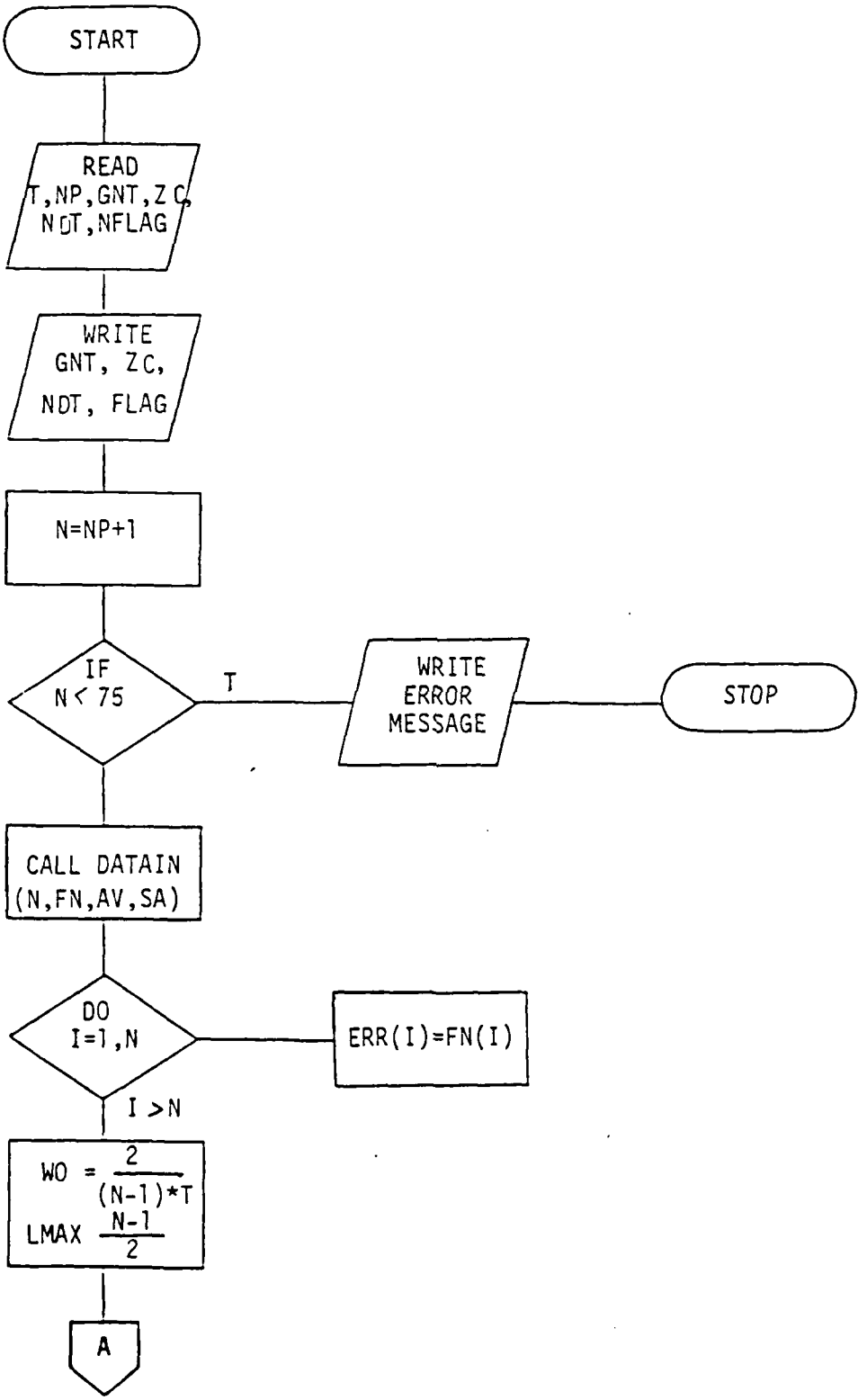
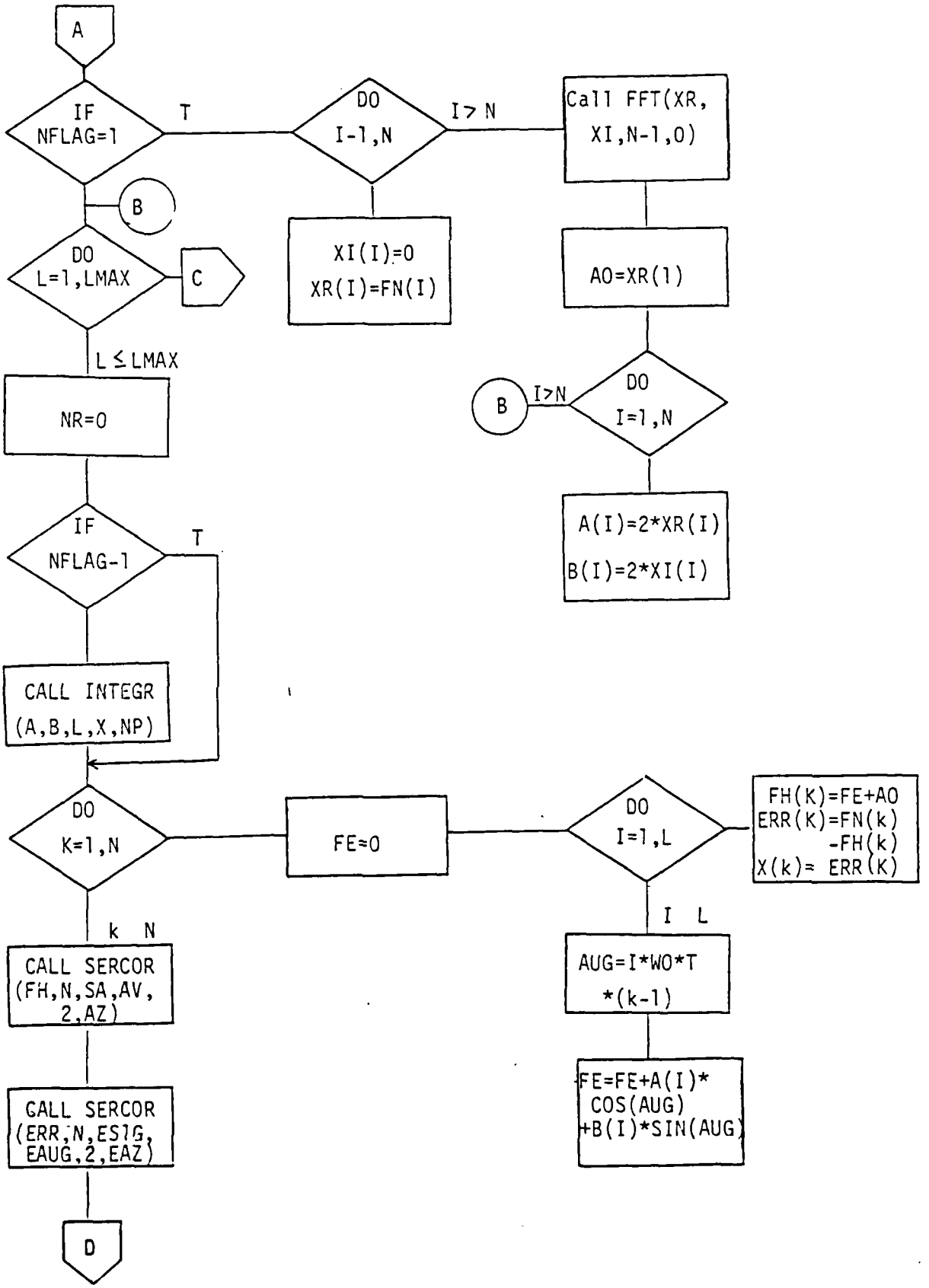


Figure 1 (Continued)



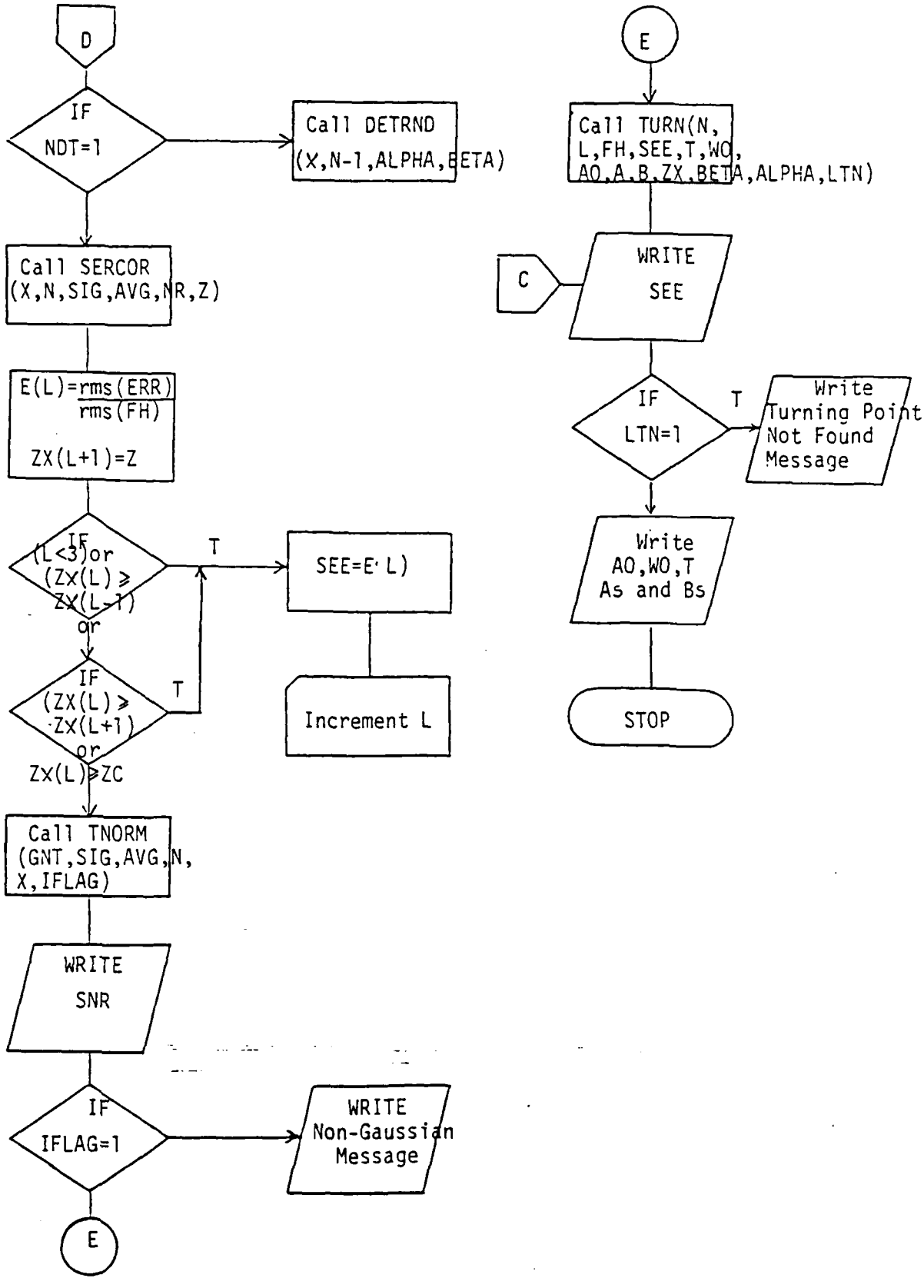
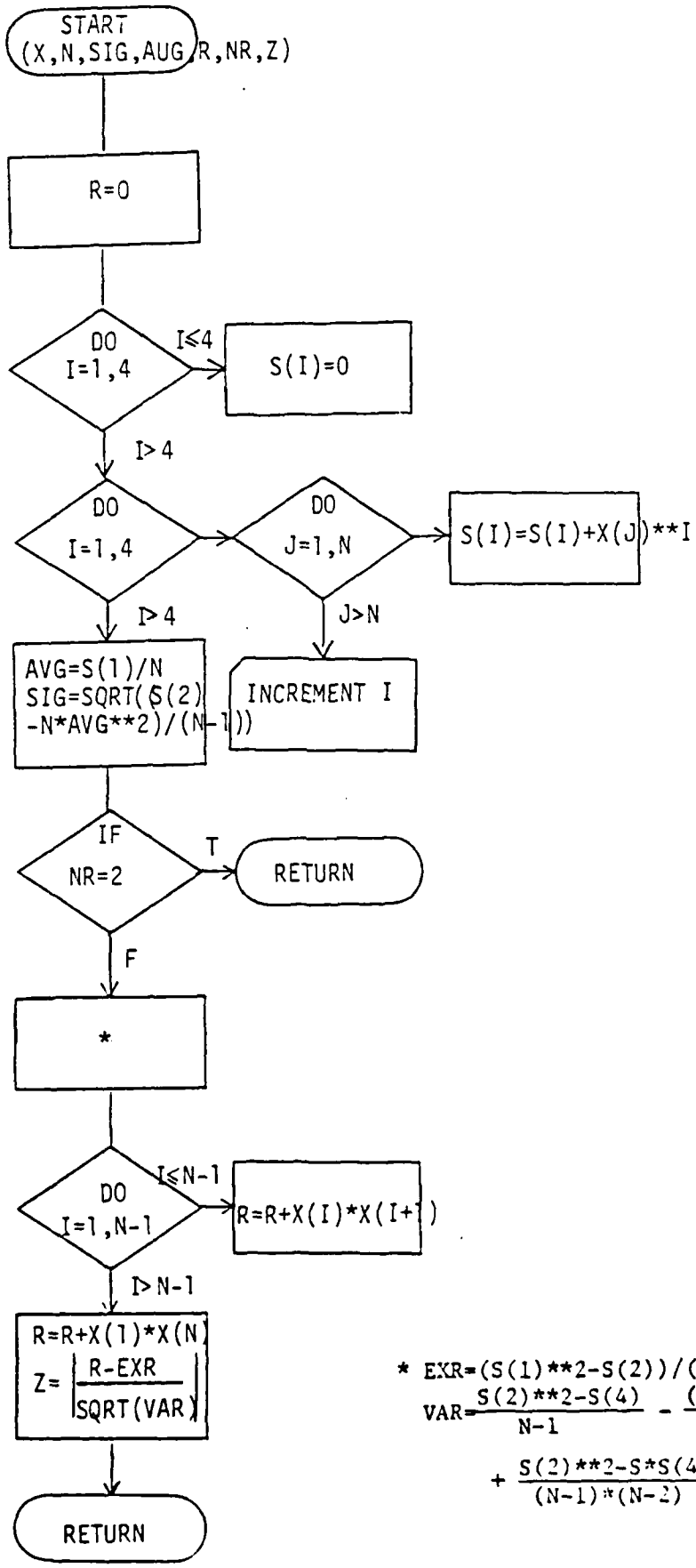

D
IF
NDT=1
Call DETRND
(X,N-1,ALPHA,BETA)
Call SERCOR
(X,N,SIG,AVG,NR,Z)
E(L)=rms(ERR)/rms(FH)
ZX(L+1)=Z
IF
(L<3)or
(ZX(L)≥
ZX(L-1)
T
SEE=E(L)
or
IF
(ZX(L)≥
ZX(L+1)
T
Increment L
or
ZX(L)≥ZC
Call TNORM
(GNT,SIG,AVG,N,
X,IFLAG)
WRITE
SNR
IF
IFLAG=1
WRITE
Non-Gaussian
Message
E
E
Call TURN(N,
L,FH,SEE,T,WO,
AO,A,B,ZX,BETA,ALPHA,LTN)
C
WRITE
SEE
IF
LTN=1
T
Write
Turning Point
Not Found
Message
Write
AO,WO,T
As and Bs
STOP

FIGURE 2

Subroutine SERCOR



* $EXR=(S(1)^{**}2-S(2))/(N-1)$

$$VAR=\frac{S(2)^{**}2-S(4)}{N-1}-\frac{(S(1)^{**}2-S(2))^{**}2}{(N-1)^{**}2}+\frac{S(2)^{**}2-S*S(4)}{(N-1)*(N-2)}$$

FIGURE 3

Subroutine TNORM

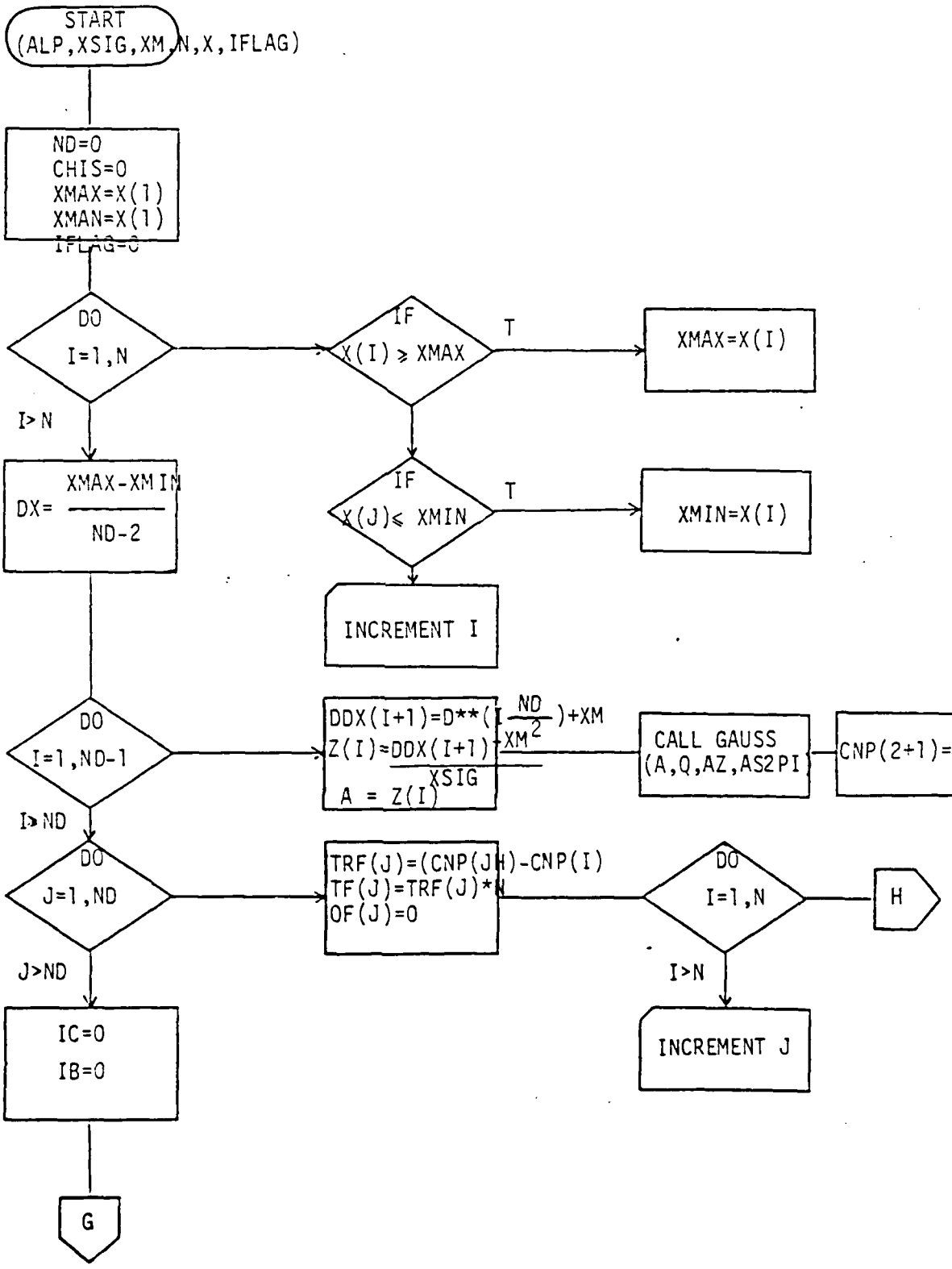
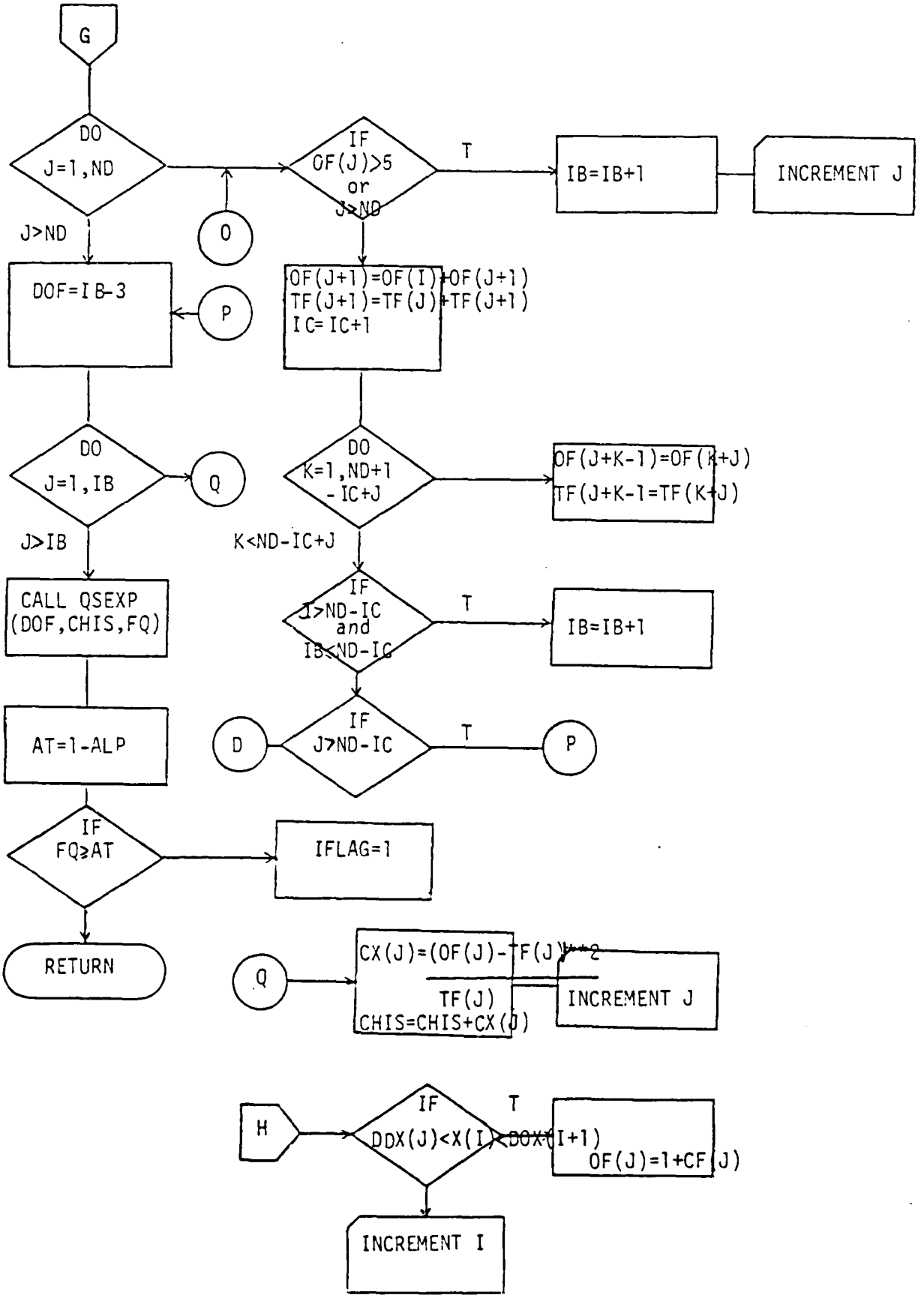


FIGURE 3 (Continued)



SAMPLE RUN STREAMS

```
ERD131$ RUN CSAF
ENTER T,NP,GNT,ZC,D-CODE,FFT-CODE

1.949147E-10,128,.05,1.96,1,0
GAUSSIAN NOISE TEST CONSTANT =0.0500
CRITICAL Z-VALUE = 1.9600
FFT-CODE = 0
D-CODE = 1

     SNR <=>     7.988648453179364      DB
NOISE IS NON-GAUSSIAN AT THE 5.0 % LEVEL.
RMS CURVE-FIT ERROR = 0.40421E+00


     AO            WO             T             N

 0.135005D+06 0.251840D+09 0.194915D-09    128

HARMONIC          A               B

      1     -0.196020D+07     -0.437464D+06
      2     -0.167392D+07     -0.130978D+07
      3     -0.144792D+07     -0.114814D+07
      4     -0.217603D+07     -0.238242D+07
      5     -0.173992D+07     -0.402266D+07
      6     -0.400472D+06     -0.681620D+07
      7      0.364314D+07     -0.951500D+07
      8      0.112735D+08     -0.298869D+07
      9      0.800790D+07      0.190004D+07
     10      0.526964D+07      0.315966D+07
     11      0.396780D+07      0.260917D+07
     12      0.309813D+07      0.253083D+07
     13      0.300151D+07      0.235267D+07
     14      0.237561D+07      0.206848D+07
     15      0.213447D+07      0.192878D+07
     16      0.212799D+07      0.186168D+07
     17      0.189028D+07      0.166967D+07
     18      0.199106D+07      0.167398D+07
     19      0.201404D+07      0.173348D+07
     20      0.189510D+07      0.178378D+07
     21      0.206599D+07      0.199034D+07
     22      0.196744D+07      0.263810D+07
     23      0.134900D+07      0.346295D+07
     24     -0.605433D+06      0.358583D+07
     25     -0.147068D+07      0.216188D+07


FORTRAN STOP
```

```
ERD131$ RUN CSAF
ENTER T,NP,GNT,ZC,D-CODE,FFT-CODE

1.,128,.05,1.96,1,0
GAUSSIAN NOISE TEST CONSTANT =0.0500
CRITICAL Z-VALUE = 1.9600
FFT-CODE = 0
D-CODE = 1

RMS CURVE-FIT ERROR = 0.86745E+00

TURNING POINT BELOW CRITICAL Z-VALUE WAS NOT FOUND.
```

AO	WO	T	N
-0.156881D-01	0.490874D-01	0.100000D+01	128

HARMONIC	A	B
1	-0.224400D-01	-0.153470D-01
2	-0.195964D-01	-0.177122D-01
3	-0.108471D-01	-0.267360D-01
4	-0.316106D-02	-0.192921D-01
5	-0.532486D-03	-0.171139D-01
6	0.173919D-02	-0.140527D-01
7	0.490572D-02	-0.124423D-01
8	0.633246D-02	-0.841825D-02
9	0.639562D-02	-0.661888D-02
10	0.745068D-02	-0.545358D-02
11	0.667389D-02	-0.308689D-02
12	0.649423D-02	-0.197764D-02
13	0.439387D-02	-0.128427D-02
14	0.506233D-02	0.476519D-04
15	0.437911D-02	0.340898D-03
16	0.266366D-02	0.442173D-03
17	0.254413D-02	0.116015D-02
18	0.103261D-02	0.106905D-02
19	0.255565D-03	0.121604D-02
20	-0.668729D-03	0.852049D-03
21	-0.187593D-02	0.830123D-03
22	-0.337533D-02	-0.139040D-02
23	-0.424310D-02	-0.227255D-02
24	-0.503819D-02	-0.392823D-02
25	-0.349550D-02	-0.570755D-02
26	-0.314827D-02	-0.467519D-02
27	-0.580768D-02	-0.502737D-02

28	-0.100394D-01	-0.121469D-01
29	-0.494963D-02	-0.199598D-01
30	0.319914D-02	-0.236230D-01
31	0.100948D-01	-0.211510D-01
32	0.153415D-01	-0.220744D-01
33	0.261958D-01	-0.670005D-02
34	0.251788D-01	-0.238465D-02
35	0.246631D-01	0.306443D-02
36	0.166233D-01	0.117785D-01
37	0.699155D-02	0.142741D-01
38	0.674609D-03	0.757396D-02
39	0.100768D-02	0.848510D-03
40	0.373425D-02	-0.343199D-03
41	0.531847D-02	-0.533471D-03
42	0.601553D-02	-0.669413D-03
43	0.623581D-02	0.884320D-03
44	0.625636D-02	0.628263D-03
45	0.603389D-02	0.116271D-02
46	0.649517D-02	0.103042D-02
47	0.636316D-02	0.129542D-02
48	0.683805D-02	0.203499D-02
49	0.599809D-02	0.276188D-02
50	0.566925D-02	0.356314D-02
51	0.498097D-02	0.335197D-02
52	0.355620D-02	0.327840D-02
53	0.282823D-02	0.288759D-02
54	0.204491D-02	0.243424D-02
55	0.811376D-03	0.123426D-02
56	0.982748D-04	-0.186041D-03
57	-0.297029D-03	-0.190078D-02
58	-0.580123D-04	-0.472191D-02
59	0.196478D-02	-0.715446D-02
60	0.503445D-02	-0.936945D-02
61	0.995390D-02	-0.878476D-02
62	0.147201D-01	-0.441118D-02
63	0.121193D-01	-0.117408D-02
64	0.162775D-01	-0.142428D-16

FORTRAN STOP

END

10-87

DTIC